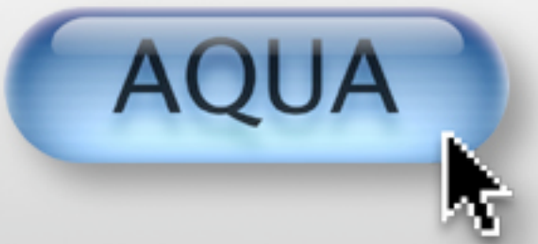
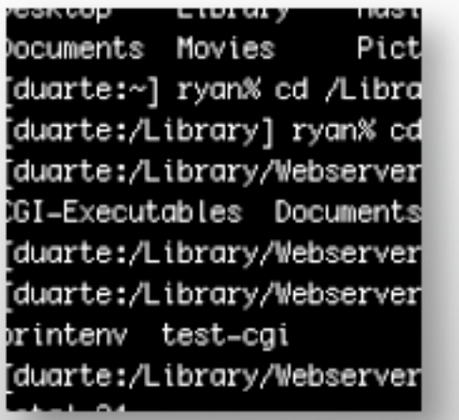
Extending Professional Applications with Plug-ins

Session 006





Extending Professional Applications with Plug-ins

Brett R. Halle
Director, Pro Video Applications
brett@apple.com

Apple’s Pro Applications

Audio Units Support

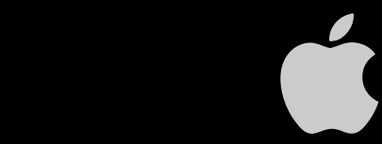
Roger Powell
Lead Audio Software Engineer, Final Cut Pro

Audio Units in Apple Pro Apps

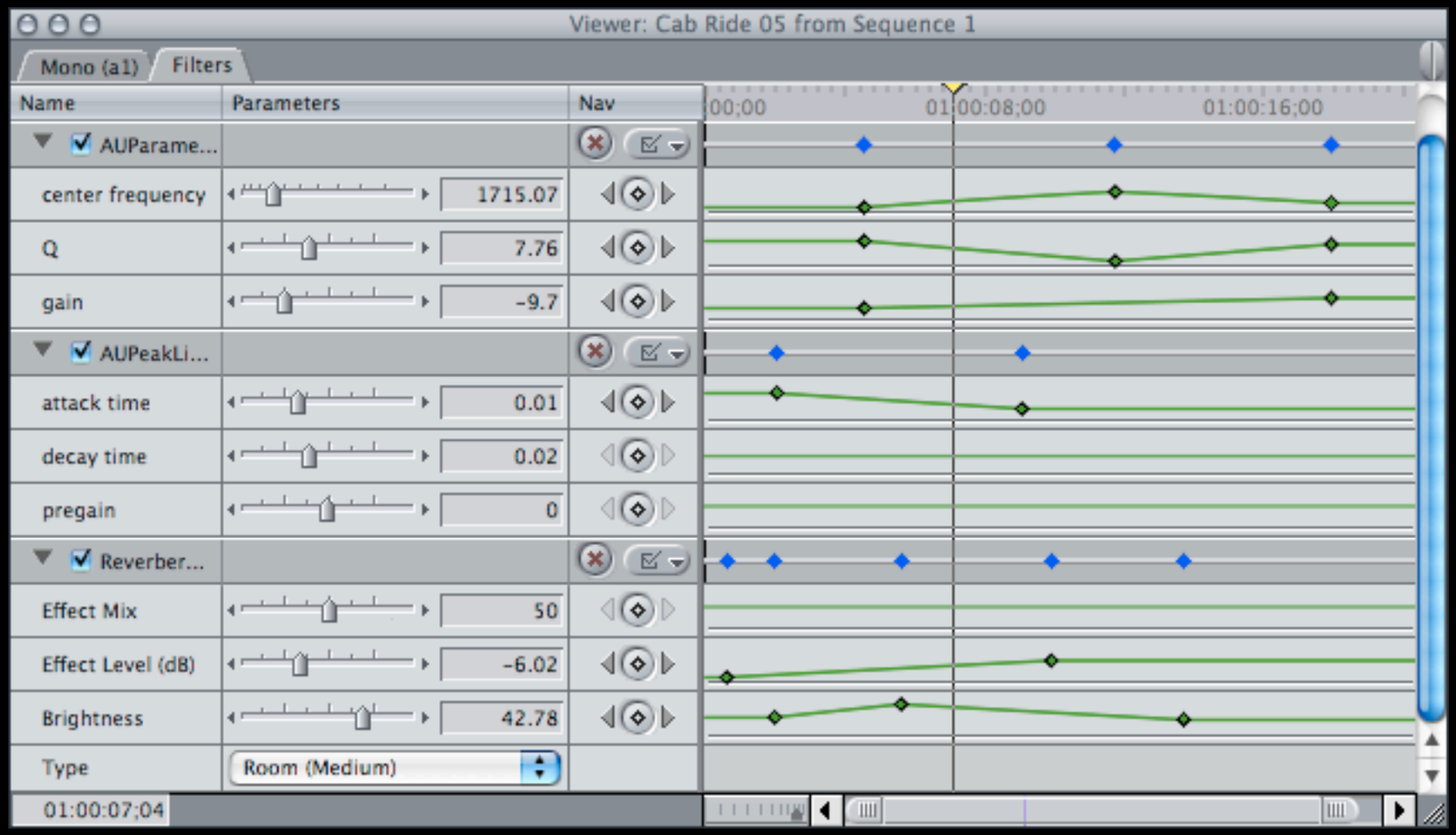
- Final Cut Pro 4.0, Soundtrack 1.0, Logic 5.5 and 6.0
- See developer.apple.com/audio for info on Mac OS X Core Audio Unit architecture
- Each application has individual requirements for:
  - Custom/generic plug-in UI
  - Input/output channel configuration
  - Audio Unit property implementations

Final Cut Pro: Audio Unit Support

- Generic FCP filter viewer UI controls
  - Custom UI not supported at this time
  - Parameter descriptions (type, range) must be complete to support UI controls
  - Plug-ins appear in audio effects folder with manufacturer's name
- Supports Mono-in and Mono-out units only
  - Plug-ins with other channel configurations not loaded
- AU presets not supported at this time
  - User settings saved/loaded via 'favorites'



Final Cut Pro: Audio Unit View



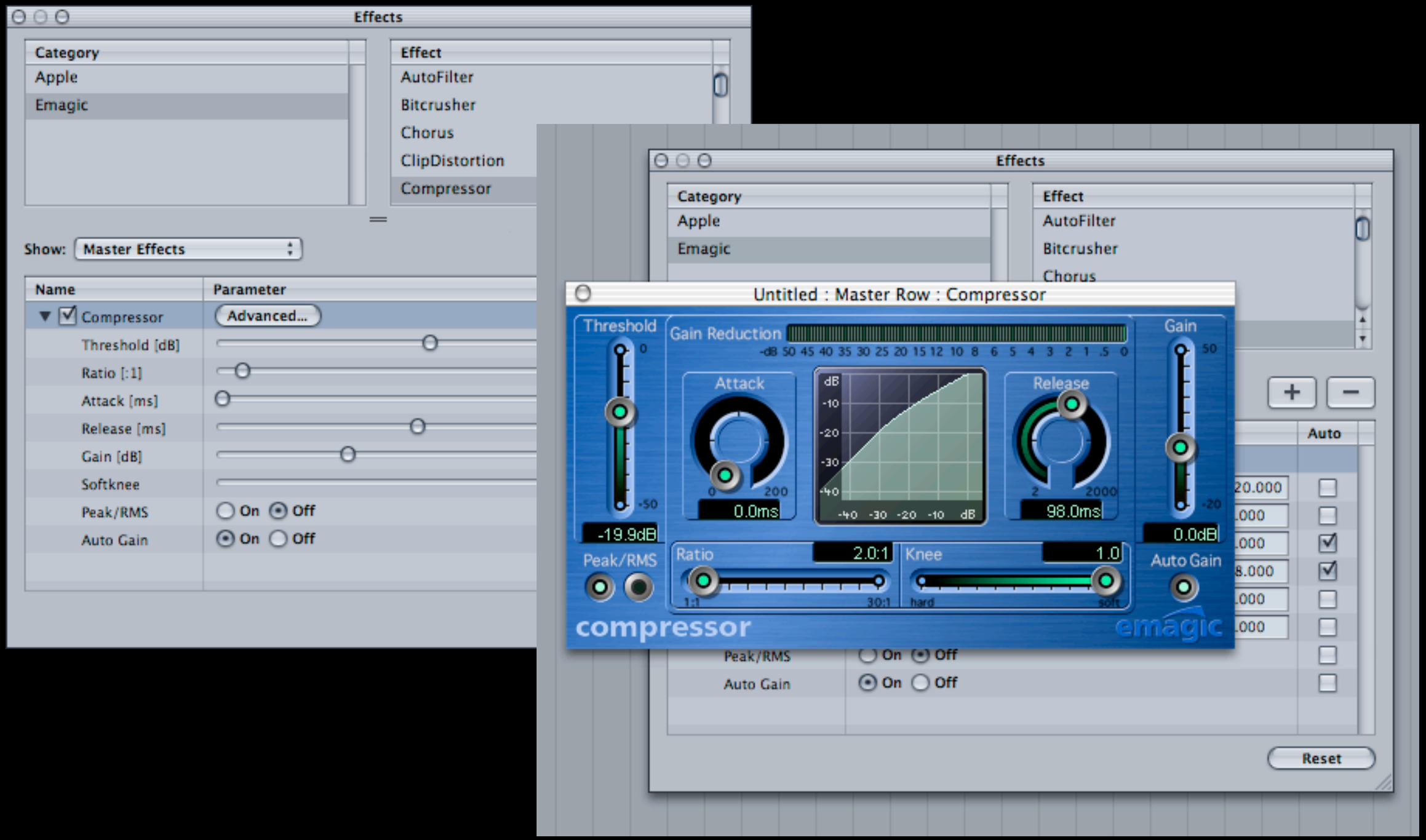
Final Cut Pro: Audio Unit Support

- Expects kAudioUnitProperty_tailTime implementation
  - Indicates length of sample history to reach steady-state
  - Used for pre-roll computation to maintain sample-accurate seaming across edits
  - If not implemented, clip items with plug-in require rendering at medium/high playback quality setting
  - Should report ‘not-implemented’ if infinite, or if state can be indeterminate (i.e., async LFO modulation)
  - Should also report latency property if applicable

Soundtrack: Audio Unit Support

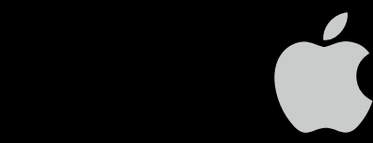
- Requires complete and accurate parameter descriptions (types and ranges)
- Supports Audio Unit Custom UIs and presets
  - Click “Advanced” button in Effect Panel
- Supports Stereo-in and Stereo-out only
- Uses kAudioUnitProperty_TailTime
  - Only required for AUs with an audible tail
  - Property value must be accurate to avoid artifacts (due to truncation of audio output)

Soundtrack: Audio Unit Views

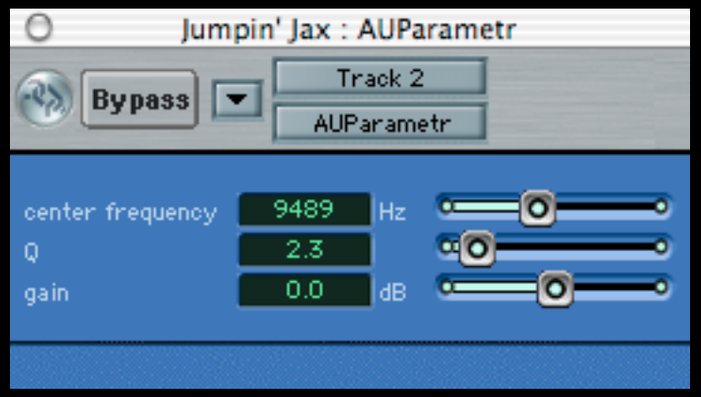


Emagic Logic Audio Unit Support

- Reference hosts are Logic Platinum 5.5.1 or higher, and Logic Platinum 6.0.1 or higher
- Put resource allocations into initialization, not instantiation for quicker start-up
- Custom UI supported
  - Generic parameter UI also supported
    - Parameter descriptions must be complete and accurate
- Supports units with Mono-to-Mono, Mono-to-Stereo, and Stereo-to-Stereo audio channel configurations



Logic: Audio Unit Views



Emagic Logic Audio Unit Support

- Uses standard ‘.aupreset’ format for save/load presets
- Supports Audio Unit latency and tail-time properties
- Plug-in should send Audio Unit mouse down/up control events to host
- Use parameter listener scheme to inform host of parameter changes

Shake Plug-in Development

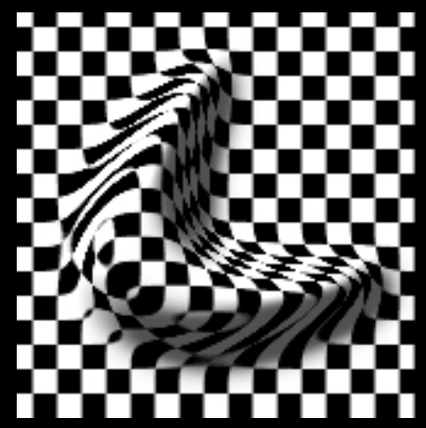
Angus Taggart
Core Technologies Engineer, Professional Applications

Overview

- Shake intro
- Who uses Shake?
- Who uses the Shake SDK?
- Shake SDK Package
- Plug-in basics

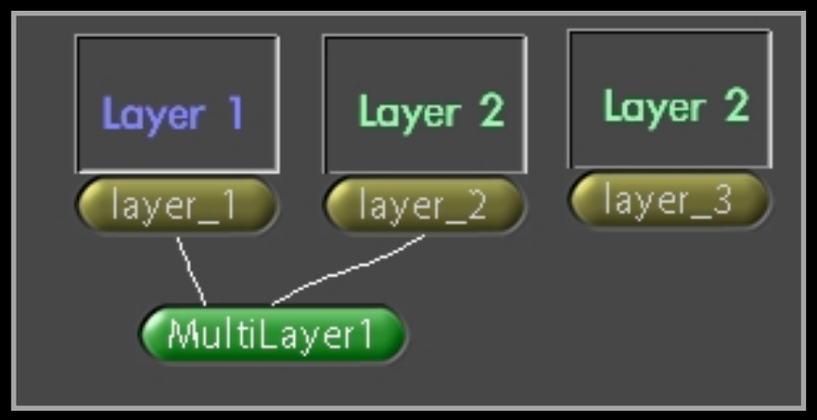
Shake Intro

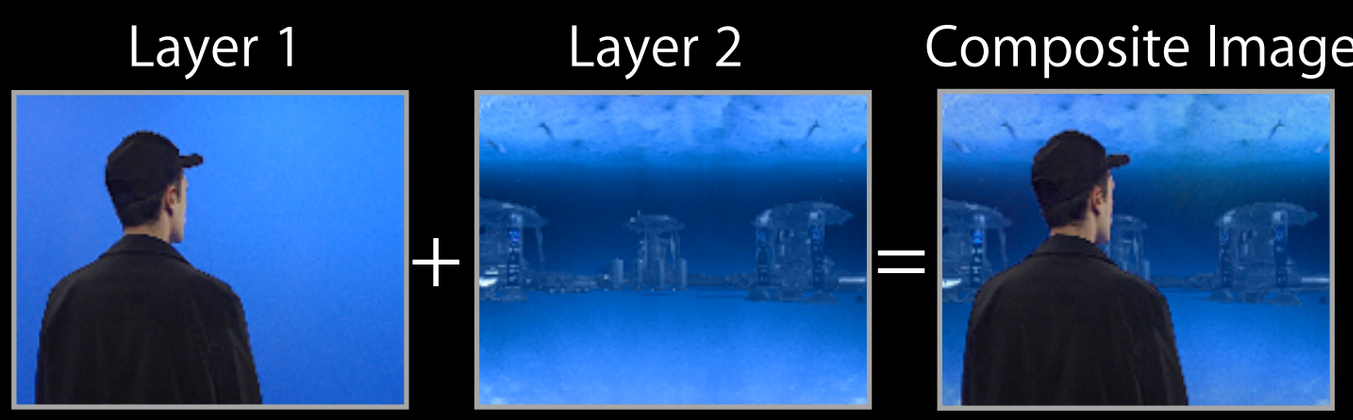
- Advanced Compositing and 2D Effects software:
  - Color correction
  - Filters
  - Keying
  - Tracking and transforms
  - Masking, rotoscoping, paint
  - (re)timing controls
  - http://www.apple.com/shake/specs.html



How Does Shake Work?

- Advanced node-based compositing engine
  - Processing occurs in nodes
  - Data moves in and out of nodes through plugs
  - Complex processing trees are easily created by connecting a sequence of nodes





Who Uses Shake?

- High-end VFX Companies: WETA, ESC, Cinesite, Dreamworks SKG, Blue Sky, Asylum,…
- Used in the last six Academy Award winners for Best Visual Effects
- Major film schools worldwide require Shake in courseware
- Increasing commercial and music video use (e.g., Mini Cooper ads)

Who Uses the Shake SDK?

- Leading third-party plug-in developers
  - The Foundry
  - GenArts
  - RE: Vision Effects
  - Ultimatte

Who Uses the Shake SDK?

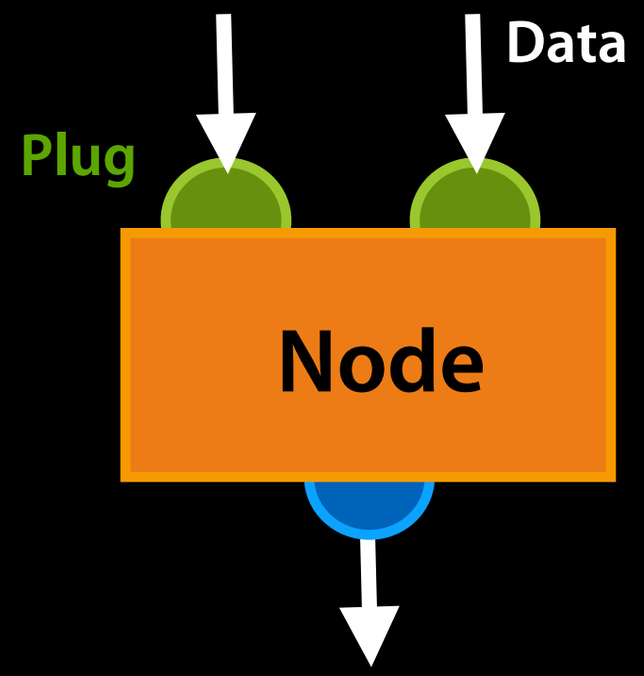
- Shake customers (high-end film production companies)
  - WETA
  - ESC Entertainment
  - Cinesite
  - Computer Film Company
- Integrate Shake into production pipelines and create proprietary FX

Shake SDK Package

- Supports Mac OS X, Linux, and Irix platforms
- Extensive Docs (ShakeSDKManual.pdf)
  - Tutorials
  - Technical overviews of Shake's Node Engine
  - Brows-able Shake class reference guide
- 18 example plug-ins
  - Image processing, widgets and overlays, retiming…
- Project Builder development environment
- Exposes Shake internal headers and frameworks

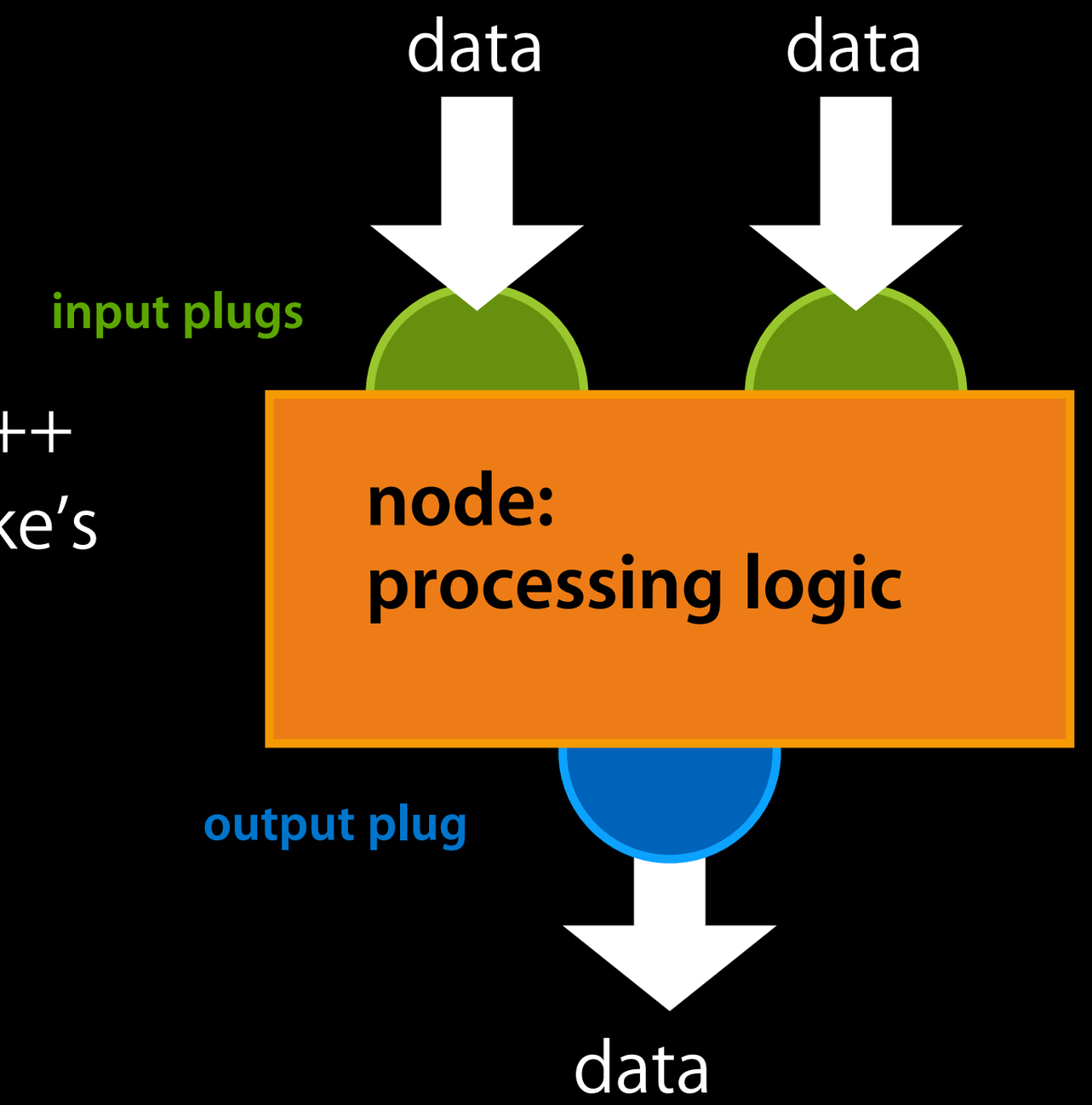
Shake Plug-in Building Blocks

- Node
  - Where processing occurs
  - Is a C++ object
  - Base class is NRiNode
- Plug
  - Where data enters and exits a node
  - Is always owned and managed by a node
  - Is a C++ object
  - Base class is NRiPlug



Mechanics of a Shake Plug-in

- A processing node in Shake's compositing engine
- Programmatically, a processing node is a C++ class derived from Shake's NRiNode base class



Node C++ Class Structure

```
class myNode : public NRiNode
{
public:
    MyNode();
    virtual ~MyNode();
    virtual int eval(NRiPlug *p);
    virtual int notify(NRiPlug *p);

    NRiPlug *_inParam1; // input
    NRiPlug *_inParam2; // input
    NRiPlug *_outValue; // output

private:
    processingMethod();
}
```

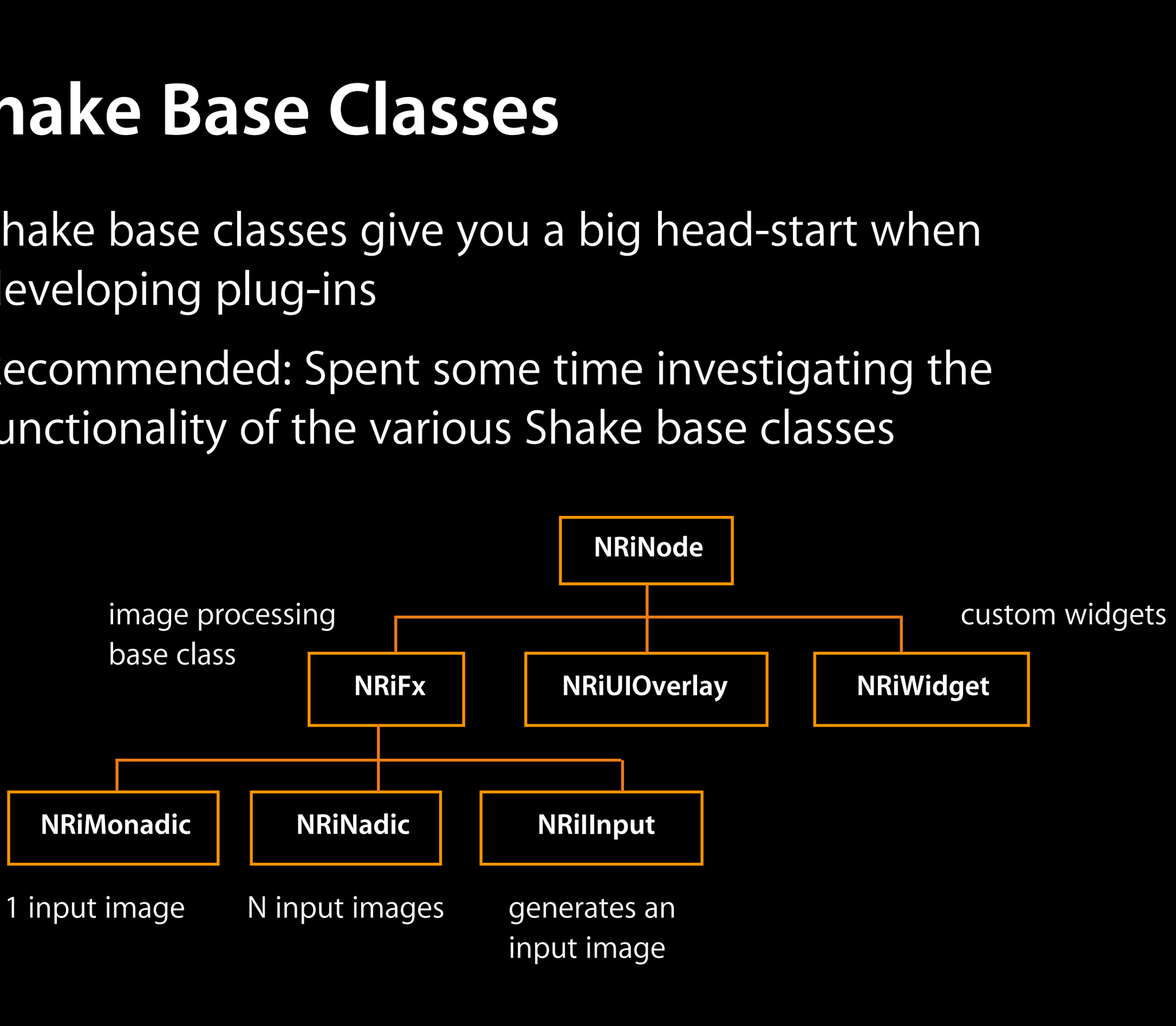
Constructor: Create plugs and internal nodes, make connections (wire the node), register dependency logic.

eval(): Compute output plug values. For this node, only the _outValue plug will be evaluated.

notify(): Handle callbacks for plugs that have been registered for notification upon change.

Shake Base Classes

- Shake base classes give you a big head-start when developing plug-ins
- Recommended: Spent some time investigating the functionality of the various Shake base classes



FXScript for Final Cut Pro

Donald Liu
Senior QA Engineer, Professional Applications

FXScript for Final Cut Pro

FXScript for Final Cut Pro

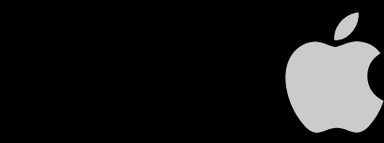
What is FXScript?

- Basic image processing programming language
- Procedural, interpreted at runtime
- Makes full use of the Final Cut Pro rendering engine
- Uses YUV, RGB, and 32-bit Floating Point processing
- All of the effects bundled with Final Cut Pro are written using FXScript

FXScript for Final Cut Pro

Language features

- Variable Types—Float, point, image buffer, string, region, and color
- Functions—Math, geometry, shapes, blit, image processing and distortion, compositing, keying, text and strings, as well as color spacing
- Input Controls—Checkbox, slider, angle, popup, radio group, color, clip, text input, and point
- Loops and Branches—If/Else, repeats, for, subroutines

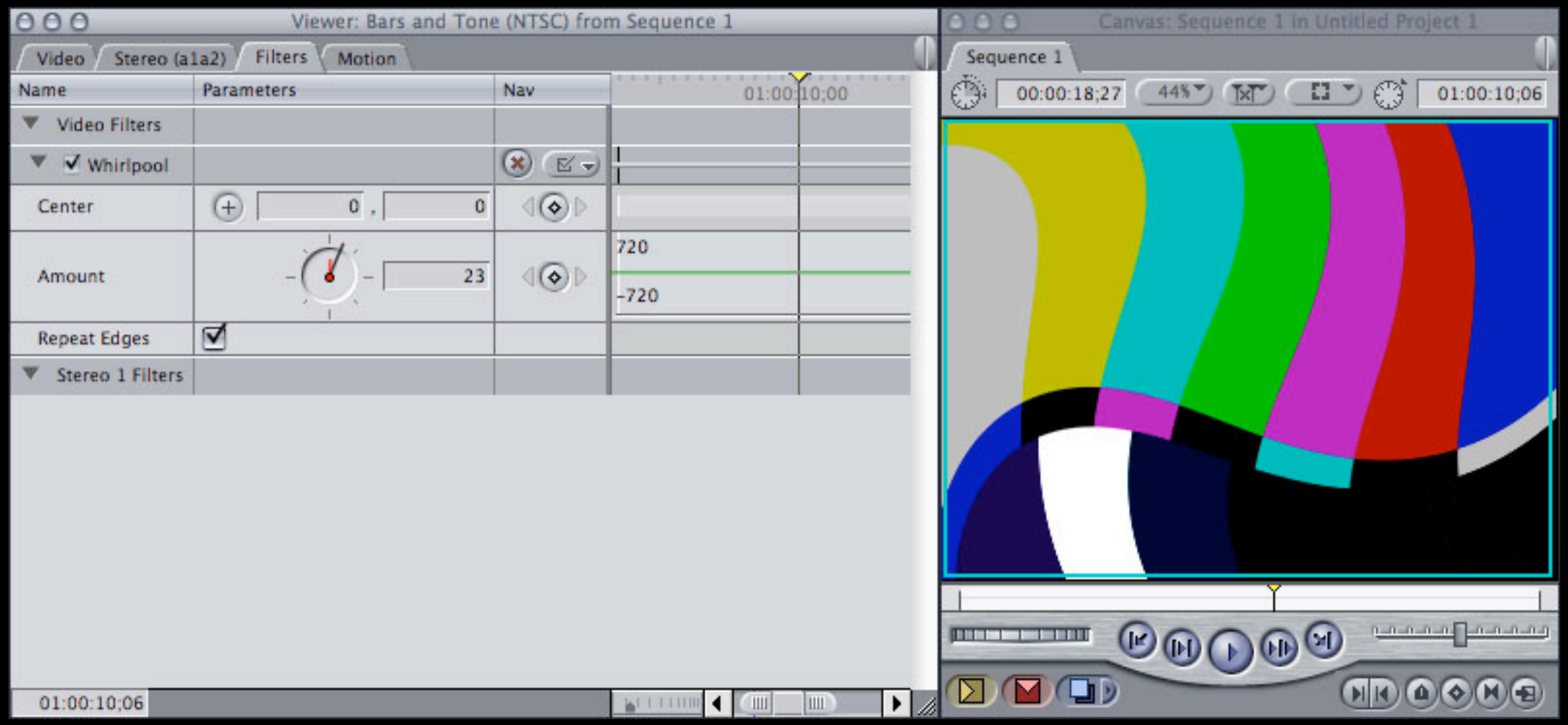


FXScript for Final Cut Pro

FXScript for Final Cut Pro

What can FXScript be used for?

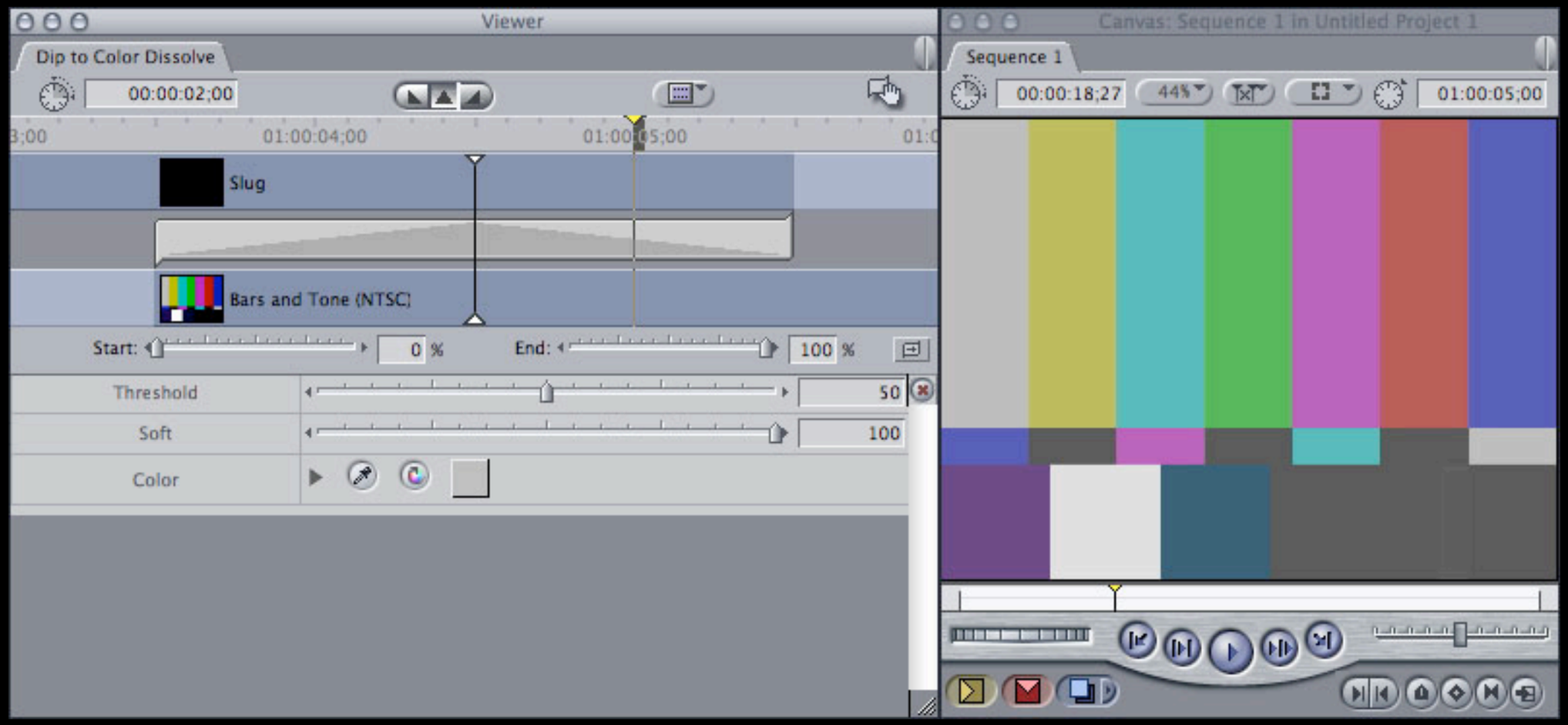
- Filters—effects that modify a video stream



FXScript for Final Cut Pro

What can FXScript be used for?

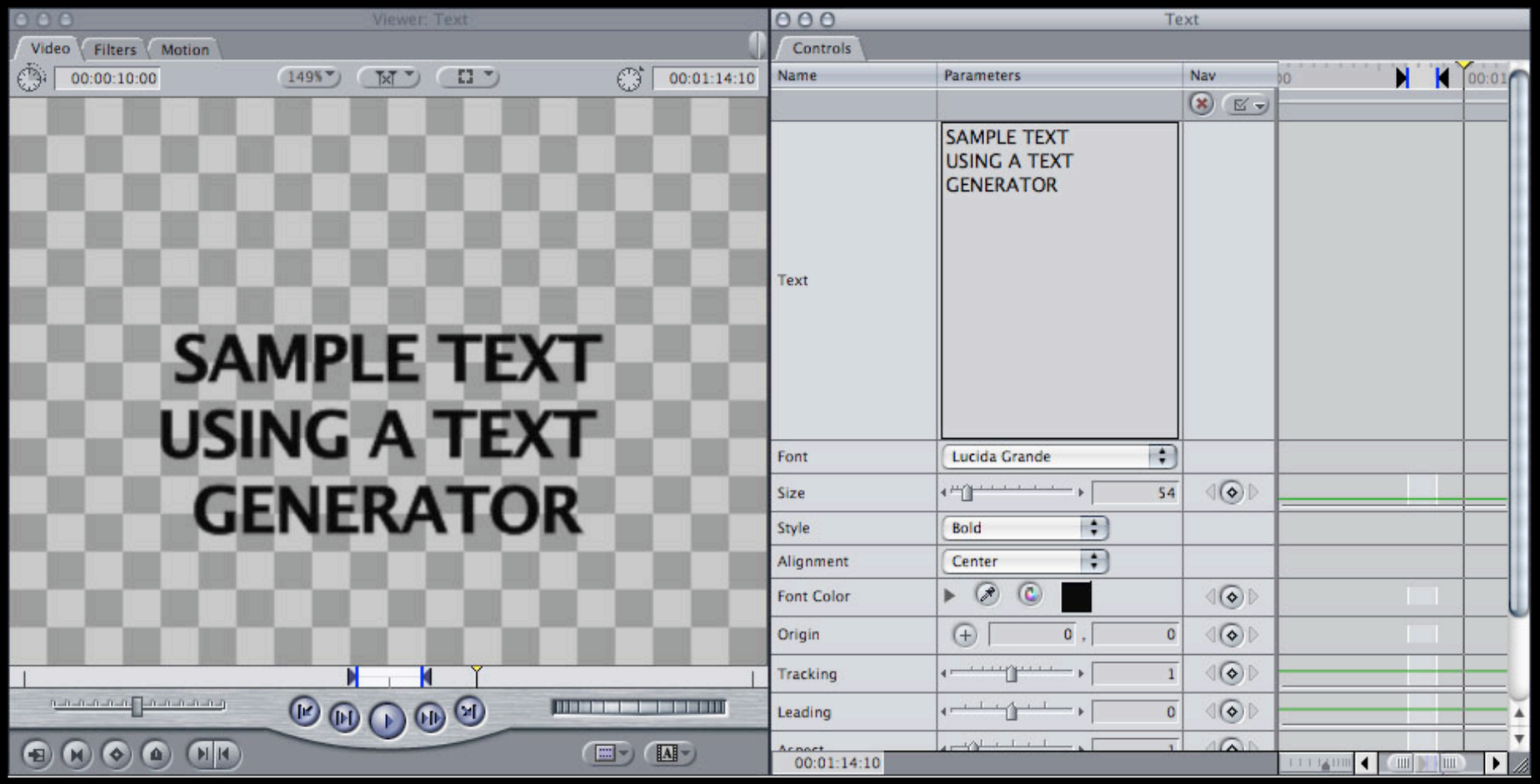
- Transitions—effects that combine 2 video streams



FXScript for Final Cut Pro

What can FXScript be used for?

- Generators—effects that generate media

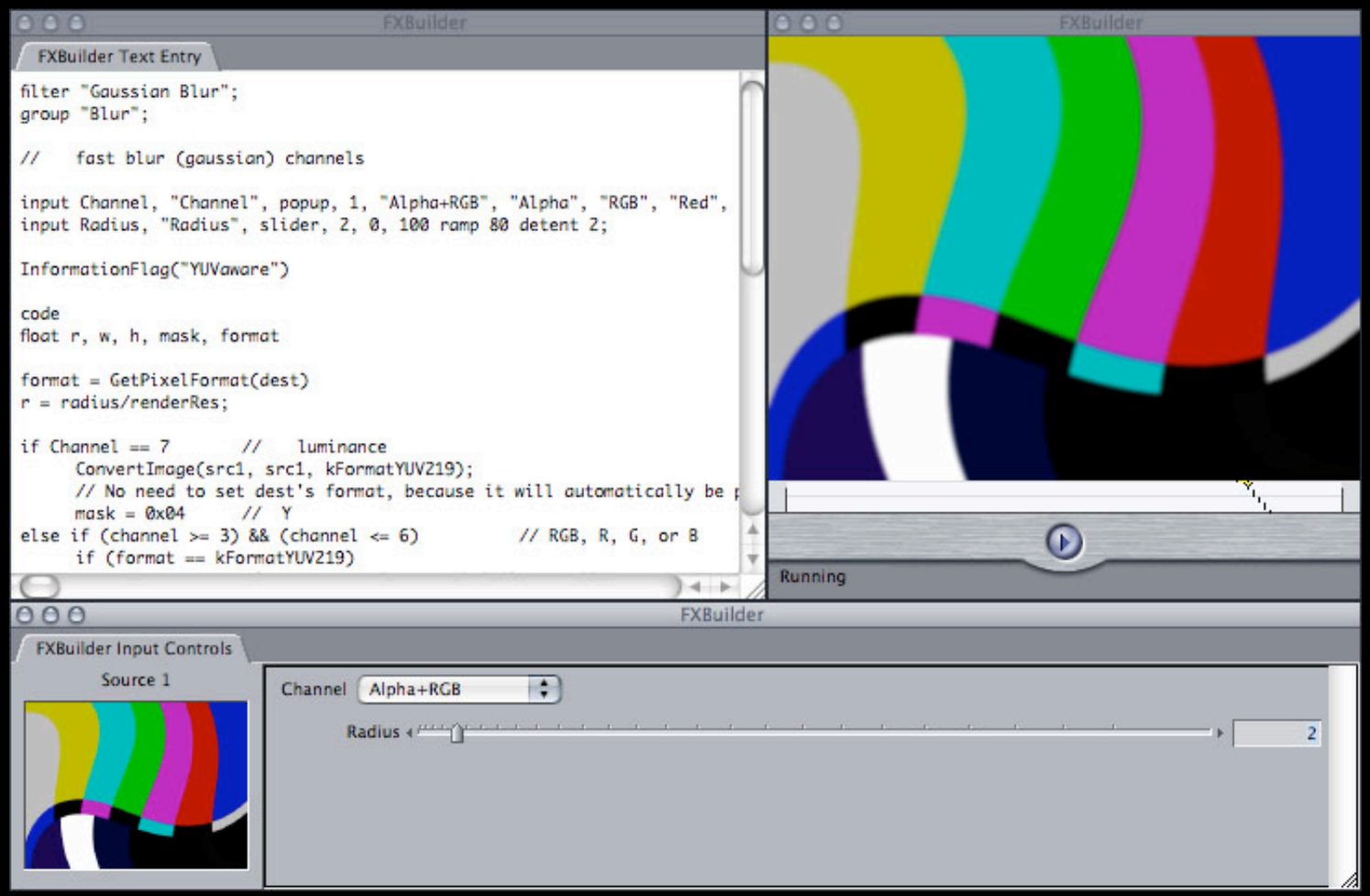


FXScript for Final Cut Pro

FXScript for Final Cut Pro

The basics: creating/editing FXScripts

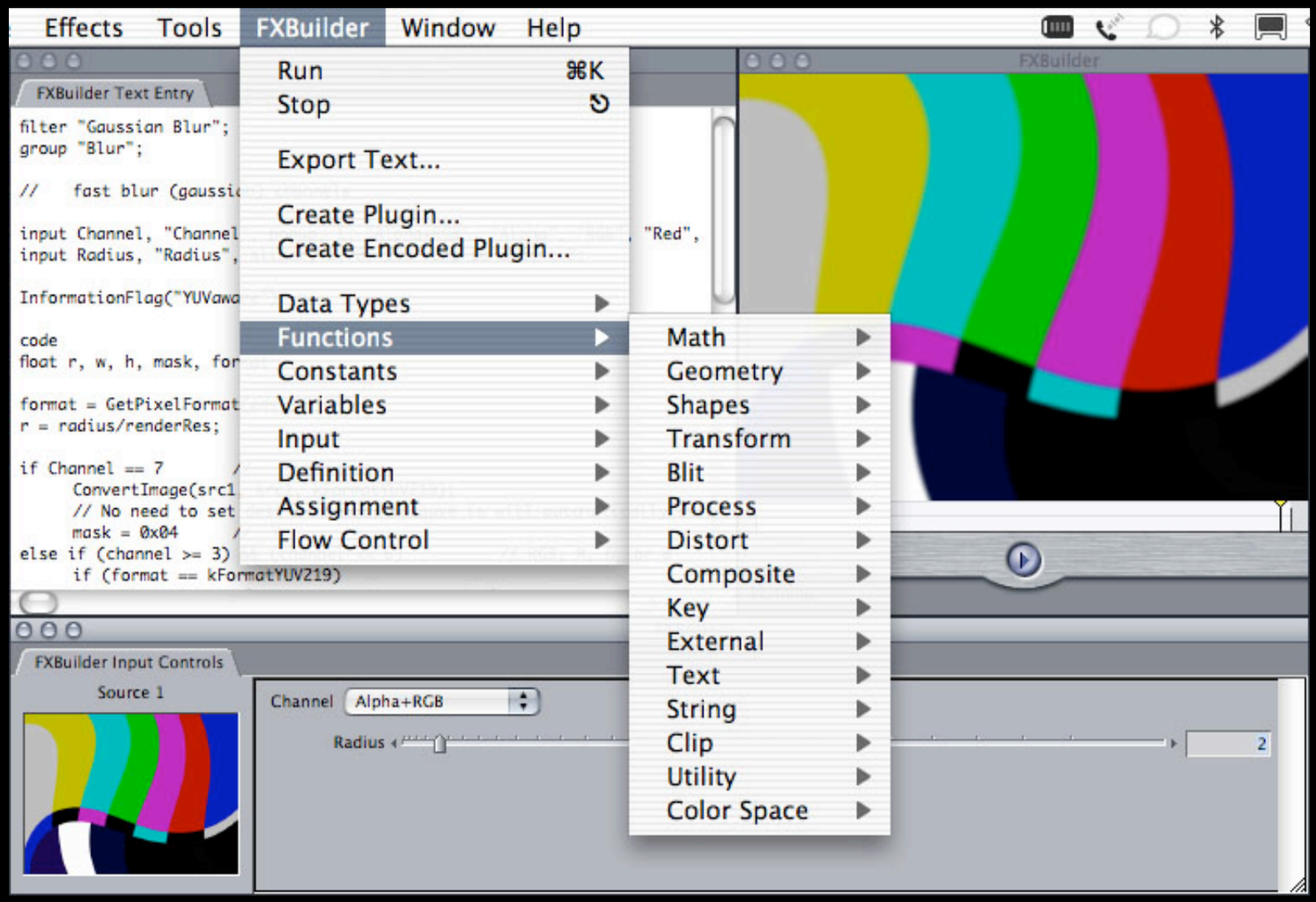
Open the FXBuilder from the Tools menu



FXScript for Final Cut Pro

The basics: creating/editing FXScripts

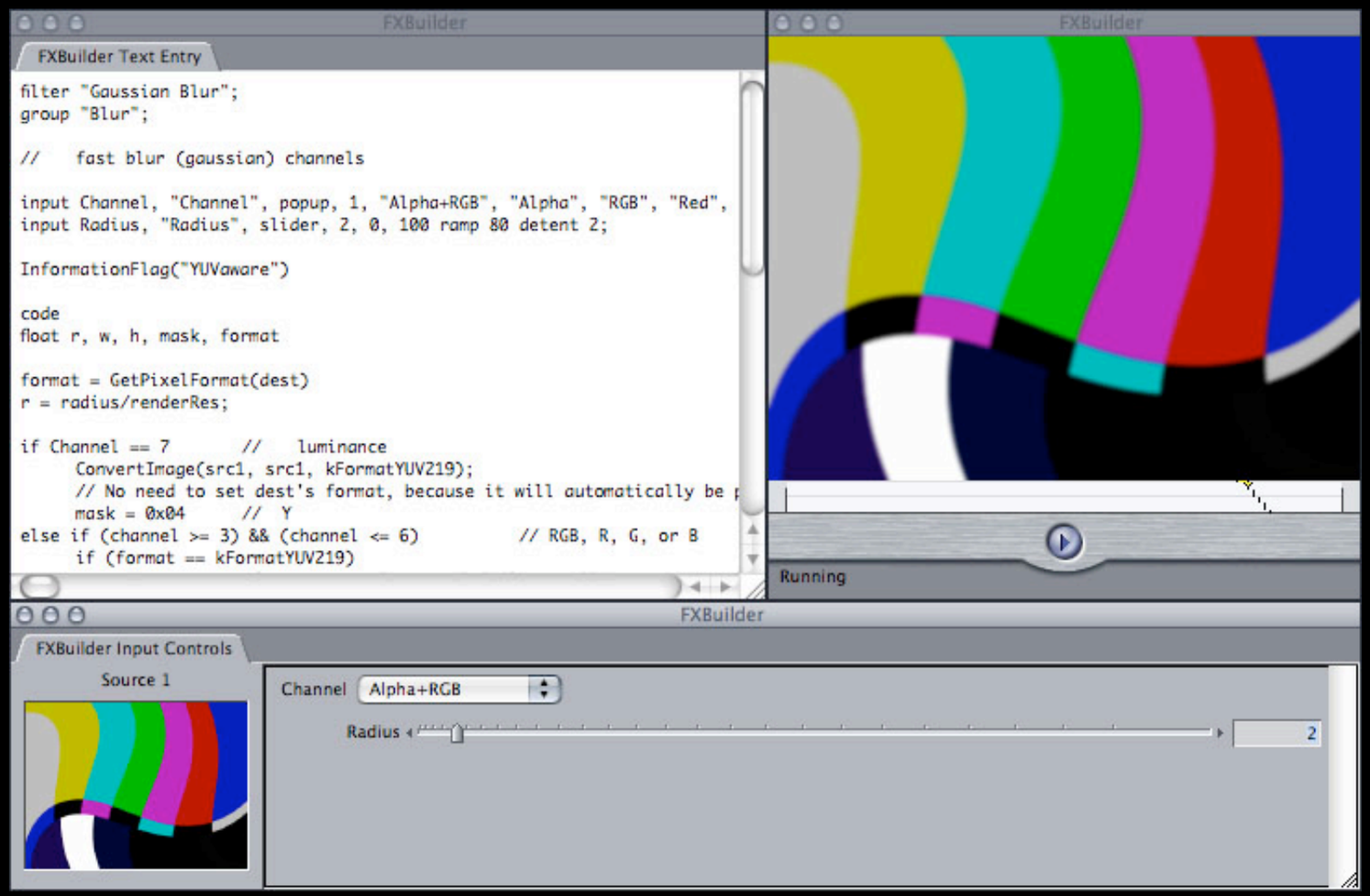
Enter your script in the text entry window, and compile it using the Run command on the FXBuilder menu.



FXScript for Final Cut Pro

The basics: creating/editing FXScripts

Adjust the parameters for your script in the input controls window, previewing the results in the viewer.



FXScript for Final Cut Pro

The basics: creating/editing FXScripts

- Save your plugin by using the Create Plugin commands in the FXBuilder menu
- All of the plugins shipped with Final Cut Pro can be viewed using the Open In Editor command on the items Browser context menu

FXScript for Final Cut Pro

Example FXScript

```
filter "Find Edges";
group "Stylize";
// blur the image, convolve two images using 3x3 Sobel's matrices,
// and select the lighter pixel of the two.

input invertdest, "Invert", checkbox, 0;
input AMOUNT, "Amount", slider, 100, 0, 100 detent 100;

code
float w, h, i, vkern[3][3], hkern[3][3], clut[256];

dimensionsOf(Dest, w, h);
image vbuf[w][h], hbuf[w][h];

// intialize Sobel operators
vkern = {-1, 0, 1, -2, 0, 2, -1, 0, 1};
hkern = {1, 2, 1, 0, 0, 0, -1, -2, -1};

blur(Src1, Dest, 1, aspectof(dest));
convolve(Dest, vbuf, vkern, 1, 0);
convolve(Dest, hbuf, hkern, 1, 0);

// select the lighter pixel from two result images
lighten(vbuf, hbuf, Dest, 1, knone);

if invertdest
   invert(dest, dest);
end if;
blend(src1, Dest, Dest, AMOUNT/100);
```

FXScript for Final Cut Pro

FXScript for Final Cut Pro

Who is FXScript useful for?

- Developers of Final Cut Pro specific effects
- Post/effect houses that need custom effects
- Advanced users looking to explore image processing

Demo

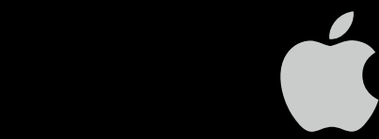
Donald Liu
Senior QA Engineer, Professional Applications

After Effects Plug-in Support in Final Cut Pro

Avi Cieplinski
Software Engineer, Final Cut Pro

Introduction

- What is After Effects?
  - Adobe's 2D and 3D compositing and effects application for video and motion graphics
- What are After Effects Plug-ins?
  - (Usually) third-party developed modules that add functionality to the host application
  - Effects, titling, keying, etc.
- Why does Final Cut Pro support After Effects plug-ins?
  - Public SDK
  - Supported by other applications



What You'll Learn

- What works
- Limitations
- Final Cut Pro specific API additions

What Works

- Final Cut Pro Supports the AE SDK v3.1
- Frame check-out and check-in
- Use of different After Effects parameter types
  - PF_Param_LAYER
  - PF_Param_POINT
  - PF_Param_SLIDER
  - PF_Param_POPUP
  - Etc.
- Plug-in defined custom parameters
  - PF_Param_CUSTOM

Limitations

- No plug-in defined on-frame UI elements
- 8-bit RGB rendering only
  - Final Cut Pro 4 has 32-bit float rendering, but not 16-bit integer
- No dynamic parameter lists
  - Final Cut Pro does not have analogous construct

Additional API Calls

- Final Cut Pro 4 provides several additional API calls that are useful for integrating plug-ins into the application
- Deal with some of the differences between Final Cut Pro and After Effects
- Useful if your plug-in has it’s own UI that is drawn over top of the Final Cut Pro UI, or if you wish to send frames out to the video out device

Additional API Calls

- There are four new calls provided in Final Cut Pro 4

```
// Retrieve the major version e.g. 4
UInt8 GetFCPMajorVersion(void);

// Retrieve the minor version e.g. x.0.1
UInt8 GetFCPMinorVersion(void);

// Force a redraw of the FCP UI
void RefreshFCPUI(void);

// Force the FCP Viewer/Canvas to a particular frame
// in the clip the effect is applied on.
// Useful if you have custom UI
void RefreshFCPPlayer(double clipFrame);
```

Additional API Calls

- Retrieve the four function pointers in a block by calling the last “reserved” function pointer in the provided PF_UtilCallbacks pointer block

```
// Function pointer block
typedef struct {
   UInt8 (*getFCPMajorVersion)(void);
   UInt8 (*getFCPMinorVersion)(void);
   void  (*refreshFCPUI)(void);
   void  (*refreshFCPFrames)(void);
} PrivateFCPCallbacks;

// Call this function to retrieve the private
// FCP callback functions for your use
PF_Err GetFCPPrivateCallbacks(void *theFunctions);
```

Conclusions

- After Effects API is a useful way to add additional image processing functions to Final Cut Pro
- Only a subset of the full After Effects API is implemented
  - Most important and appropriate base subset for Final Cut Pro to include
- We want your feedback!
  - Suggestions on other API elements to support are encouraged

Future Plug-in Directions

David R. Black
Lead Software Engineer, Final Cut Pro

Future Plug-in Directions

Unified plug-in model for professional applications

Future Plug-in Directions

Unified plug-in model for professional applications

- Supplements existing operating system plug-in models
- Common base architecture
- Function/task-specific APIs
- Single plug-in supported by multiple applications
- Multiple languages and development environments

Future Plug-in Directions

Unified plug-in architecture

Future Plug-in Directions

Unified plug-in architecture

- Based on Objective-C protocols
- Packaged as standard bundles
- Static and/or dynamic plug-in loading and registration
- Host application data and services accessible using objects

Future Plug-in Directions

What about other plug-in models?

Future Plug-in Directions

What about other plug-in models?

- Unified plug-in model, combined with function/task specific APIs, is intended to be a superset of existing plug-in APIs
- Designed to support adaptor/host plug-ins
- Intention to provide adaptor plug-ins for the most popular formats as the market dictates

Future Plug-in Directions

When will the unified plug-in model be available?

Future Plug-in Directions

When will the unified plug-in model be available?

- Final Cut Pro 4 will be the first application to support the unified plug-in model
- Data interchange plug-ins (using XML) will be the first functional API implemented
- Available to developers in a public beta in August
- Final release to end users and developers later this year

Summary

Plug-In Support Summary

Plug-In Support Summary

All of Apples' professional applications support one or more plug-in models

Plug-In Support Summary

All of Apples' professional applications support one or more plug-in models

- Logic, Soundtrack
  - Audio Units
- Shake
  - Shake Plug-ins
- Final Cut Pro
  - Audio Units, After Effects, FXScript

Plug-In Support Summary

Plug-ins are an important to end-users and Apple

Plug-In Support Summary

Plug-ins are an important to end-users and Apple

- Allows for new and unique tools to be developed
- Provides users more control over their end product
- Advances the technology behind applications

Plug-In Support Summary

More extensive support for plug-ins in the future

Plug-In Support Summary

More extensive support for plug-ins in the future

- Professional applications will support more plug-ins
- Movement toward modern plug-in architectures
- Ability to share plug-ins across multiple applications
- More extensive plug-in APIs to enable further enhance and extend professional applications

Plug-In Support Summary

Input from developers is extremely important

Plug-In Support Summary

Input from developers is extremely important

- What kinds of APIs are needed?
- What are your preferred development tools?
- Are there any desired tools that are not supported by existing APIs?

Wrap Up

Who to Contact

Jeff Lowe
Industry Evangelist for Professional Film and Video
jefflowe@apple.com

Final Cut Pro Feedback
Bug Reports and Feedback for Final Cut Pro
fcpfeedback@apple.com

http://developer.apple.com/wwdc2003/urls.html

Q&A

http://developer.apple.com/wwdc2003/urls.html

WWDC 2003

Worldwide Developers Conference